The Competing Project Management Methodologies

How the Specter of War and a Ski Trip Changed the Way we Deploy Software

There is little controversy or argument to be made that Agile has been the preferred approach for software development and deployment. Posed as a question as to the right methodology for an ERP, WMS, TMS --- any acronym implementation and the answer 'Agile' will likely be met with nods of approval around the room. The name alone evokes the image of an athlete deftly working around defenders in a dazzling display of control and confidence. Compare that to the suggestion of a waterfall approach. That association suggests uncontrollably careening toward a massive drop-off, uncertain if the barrel you're tucked into will protect you from a watery grave. Not exactly something that inspires confidence when taking on a significant investment of time, effort, and capital. Waterfall has the association of the 'old guard' way. Something of an antiquated approach that was done before we understood how human beings operate any large-scale project effort.

Is this conventional understanding a fair one? Do project organizers and approvers need to make a deliberate declaration to follow one or the other as a prerequisite for success? A more thoughtful review of both 'Waterfall' and Agile methodologies shows nuance, and that both schools of thought can lead to successful outcomes for a myriad of supply chain project types.

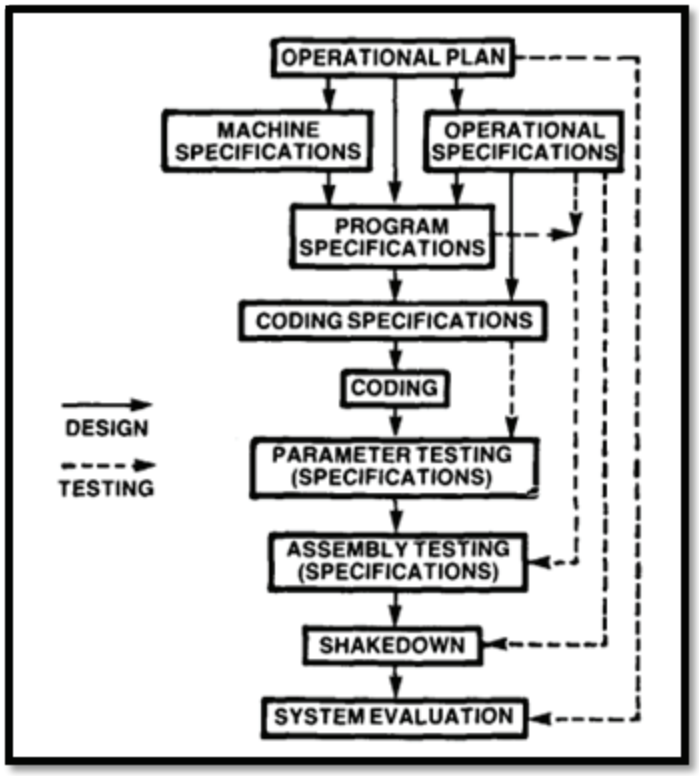
Understanding how both can be applied requires context on when and how the approaches became popular. First, the 'Waterfall' methodology.

History of Waterfall

The 'Waterfall' methodology does not have a clear consensus on its origin point. While phased-efforts date back hundreds of years around civil engineering projects, modern project methodology materials commonly point to Herbert D. Benington, Dr. Winston Royce, or some combination of the two as the Prometheans of the 'Waterfall' approach. Benington's contributions date back to 1956. In the years preceding, Benington was working in MIT's Lincoln Laboratory as a computer programmer. Computer programming was in its infancy, and he identified four recurring problems around the projects that enabled programming: computer operation, program/system reliability, supporting/dependent programs, and documentation. The stakes of addressing these challenges were high for the lab. Huge swaths of U.S. airspace had to be digitally translated from radar sites and aggregated into a single output as an early warning indicator of a potential nuclear-powered air attack by the Soviet Union. That massive effort to pull together data from disparate radar sites was the goal of the SAGE program, and the behemoths of early computers and their underlying program instructions were the engines that drove toward that objective.

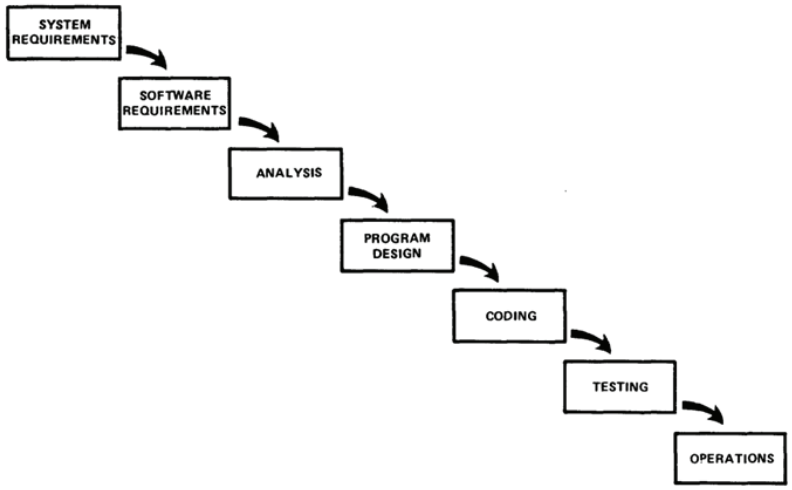
Where others failed, Benington and his colleagues at Lincoln Lab succeeded in keeping computer programming projects on track for timelines and budgets. Benington attributed his success to a repeatable approach in a 1956 Navy Mathematical Computing Advisory Panel symposium.

One of the diagrams that brought texture to his ideas, shown below, illustrates the steps that he and colleagues took to reliably address the recurring issues they encountered. The idea of execution of steps in sequence and the resemblance that his illustration bears to a waterfall provides justification to the view that Benington's work is the origin of Waterfall project methodology. Yet he never makes any mention of the term waterfall. Furthermore, the specificity of national defense and the decades-old material leaves room for competing viewpoints.



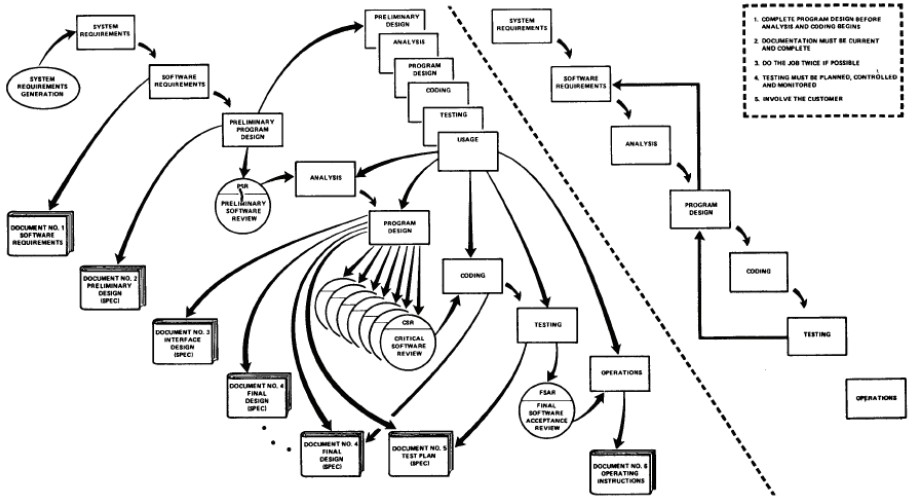
Enter Dr. Winston Royce. In 1970, the Cold War was still a threat to turn nuclear hot, but the tensions took on a competitive outlet—most notably being the Space Race. That year, Royce authored "Managing the Implementation of Large Software Systems", which captured his findings in nine years' experience on spacecraft mission planning, commanding, and post-flight projects. Royce explicitly characterizes his findings as anecdotal and developed from a biased point of view—not an equation that can be reliably applied to every situation.

In his paper, he creates a simple illustration moving from system requirements, through milestones of coding, testing, and operations – a framework which could reasonably be called the 'Waterfall' approach.



Similar to Benington, Royce never describes his methodology as ‘Waterfall’. Moreover, he continues to call out the tremendous risks that one might experience moving from one stage to the next without reverting back to a previous set of activities based on stakeholder feedback.

Royce arrives at a significantly more complex illustration through iteration in his paper, showing sub-activities, diverging paths, and conditional next steps.



In his first illustration, Royce introduces one of the clearest visual representations of the simple ‘Waterfall’ idea that many associate with conventional software development methodology and emphasizes its weaknesses at length. Thirty-years before the agile manifesto was written, Royce documented his favor towards testing, the direct relationship with customer stakeholders, and the value of iteration.

Other theories on where the exact term ‘Waterfall’ originates include an academic work by T.E. Bell and T.A. Thayer, another theory points to a Department of Defense published standard of methodologies, and another still is in a story captured between a director of IT and a prospective salesperson pitching project methodology. Regardless of the true origin of the coined term ‘Waterfall’, the modern interpretation of its rigidity as a methodology are misunderstood or deliberately overemphasized as a means of creating a contrasting viewpoint to Agile methodology.

History of Agile

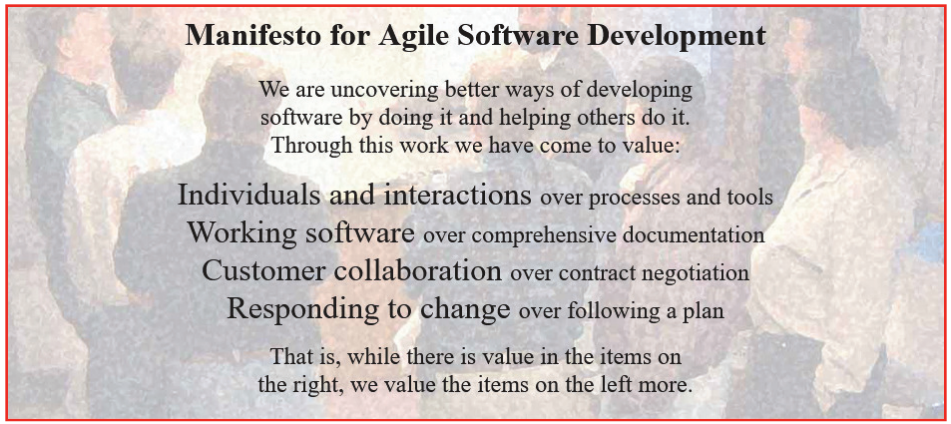
The origin story of Agile is much less ambiguous. In the fall of 2000, leaders in software development sought to set aside a time and place to discuss challenges in software development and how to overcome them. The ‘conference’ took place in February of 2001 at the Snowbird Ski Resort in Utah with seventeen individuals in attendance. Beyond the simple pleasure of a vacation with like-minded individuals, the Seventeen met to find common ground on a more effective way to develop and deploy software.

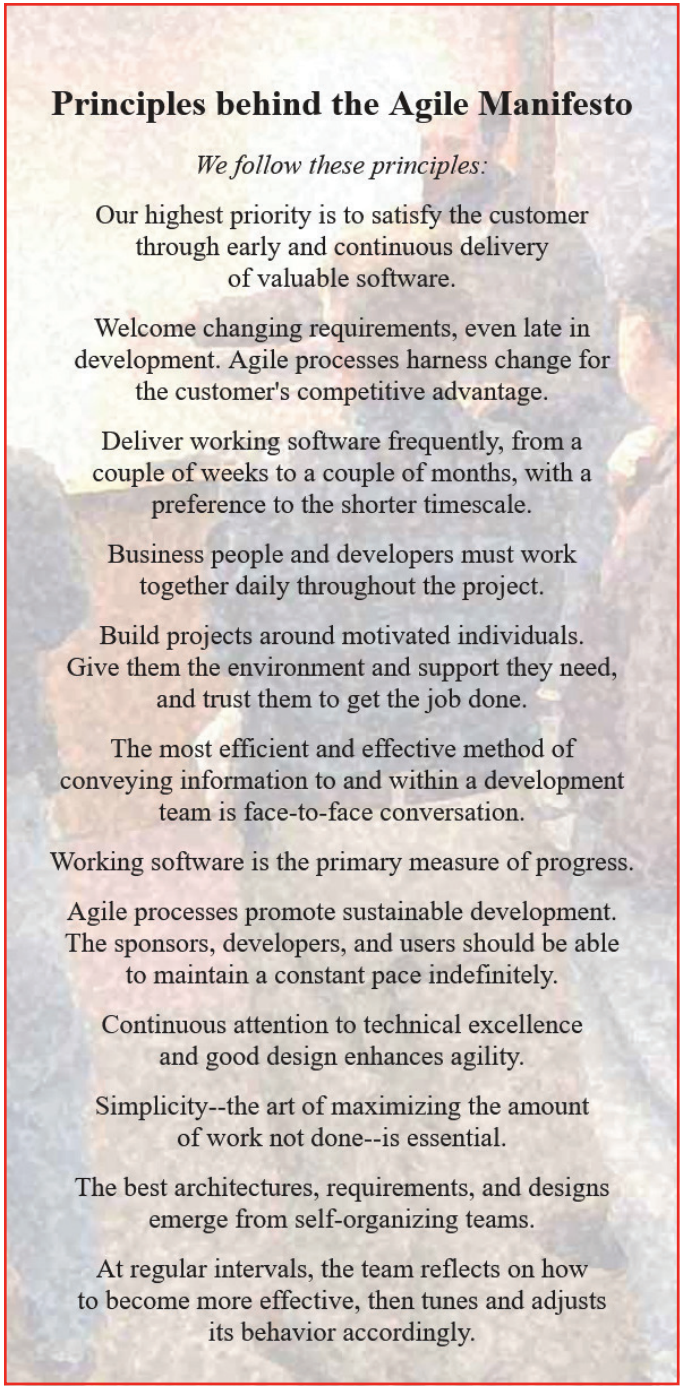
Numerous approaches to software development which marketed themselves as a counter to ‘Conventional/Waterfall’ were already well documented and used in practice.

Extreme Programming, SCRUM methodology, Dynamic Systems Development Method (DSDM), Rapid Application Deployment… all were developed or enhanced by those in attendance. It stands to reason that the original goal of the weekend meeting was to create and document a uniform methodology that would replace or combine those previous approaches.

Two days of discussion led to the Agile Manifesto, which is preserved to this day. All seventeen signed the Agile Manifesto in solidarity, despite participants commenting that there was lively discussion and some points to which the group could not agree. Remarks on the weekend from some of the participants commonly had themes of "not what was originally expected but pleased with the outcome." So what was that outcome?

It's much simpler than one might expect given the content around it today. The manifesto is comprised of a mission statement accompanied by four points of emphasis. These points of emphasis are supported by twelve principles that highlight the themes that enable successful and healthy software projects.





That is it. Nothing in the original Agile manifesto discusses a sequence of steps that can be followed for a project. Nothing about when or how to gather requirements, no specific instructions for testing formats or targeted outcomes, no series of checks to reduce risk when promoting code to a production environment. Any visual representation of Agile as a methodology is a mere interpretation of these principles. Put simply, the Agile Manifesto tells nothing about what to do in a software development or deployment project. But what if it was never intended as something to do? What if its intent was something to be? Regardless of the original intent, Agile has grown into a methodology that can be done.

Considerations on Agile and Waterfall as Competing Methodologies

Comparing the two best-known source materials for conventional/waterfall project management methodology and the original material for the Agile Manifesto shows there is nothing inherently contradictory about them. However, over time the interpretation of these principles has evolved. Each of the principles is now backed by their unique and widely-agreed upon tools and techniques. The recency of Agile's ascendance has resulted in it becoming thought of as the default project methodology. A more nuanced take is considering them as two impactful approaches which can be incorporated into modern day projects based on the project type, deliverables, resources, etc.

Those impacts are not identical for all project types. Some lend themselves to applying one approach over another inherently. Other project types can result in successful outcomes by using either project methodology. Project stakeholders benefit in understanding what is gained and lost by focusing on one approach over another.

A brief working definition of strategy, analysis, proof of concept, and implementation projects set the stage for evaluating whether an Agile or Waterfall approach will increase chances of successful outcomes.

Strategy: Defines long-term goals and roadmaps to optimize supply chain performance, align with business objectives, and respond to market dynamics. A higher-level viewpoint within and/or among the supply chain functions of planning, sourcing, manufacturing, transportation, and distribution.

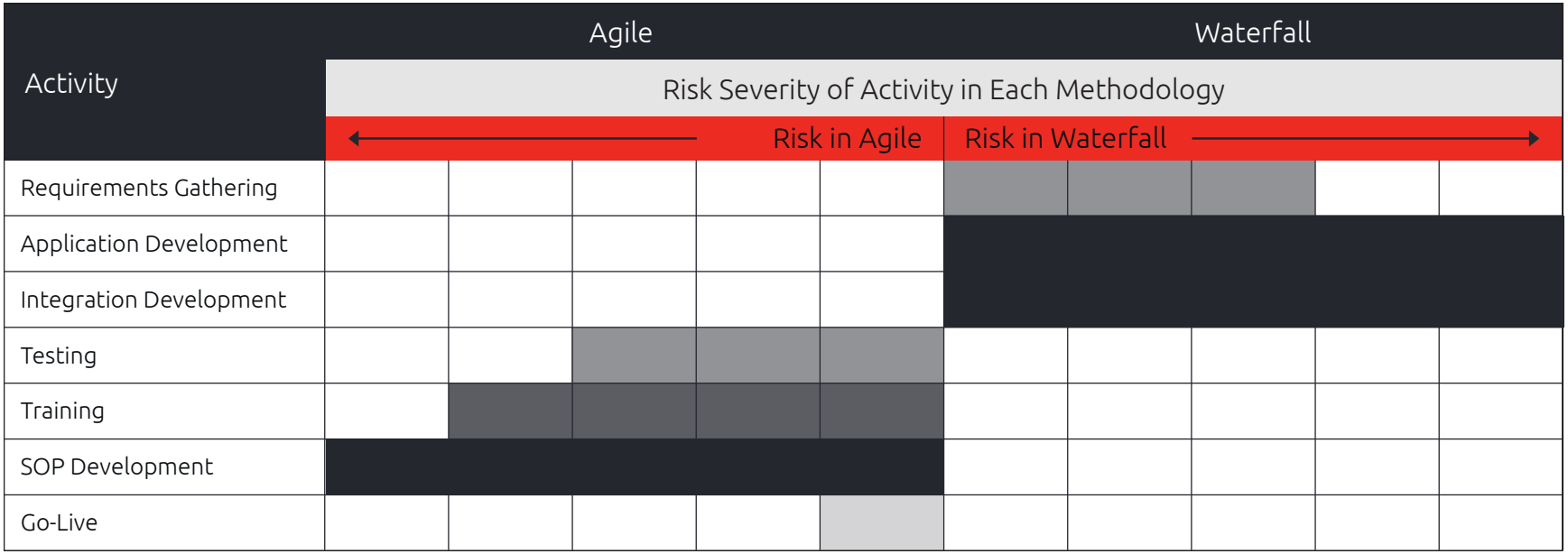
Analysis: Involves deep dives into data to uncover inefficiencies, identify cost-saving opportunities, and support decision-making through modeling and benchmarking.

Proof of Concept: Tests a targeted supply chain solution or technology on a small scale to validate feasibility, performance, and business value before broader deployment. Note – this is not always a required prerequisite step to implementation, however justified in high-risk or generally uncertain environments.

Implementation: Executes the rollout of new systems, processes, or technologies across the supply chain, ensuring alignment with requirements, user adoption, and operational continuity.

Important to emphasize is that no project methodology is a one-size-fits-all approach that guarantees success. Even within an organization, the best project methodology approach may change depending on the supporting resources or the subject matter. Still, intentionally choosing an approach establishes clarity and focus for those involved. The differences between agile and waterfall apply to the entirety of a project for the list above, with the exception of a software implementation. Within a software implementation, the differences are more apparent when comparing the underlying activities.

Project Type	Management Methodologies	
	Agile	Waterfall
Strategy	Agile is a risky or potentially inappropriate approach for true strategy development. It opens the door to constant re-assessment of the problem statement, creating the proverbial "paralysis by analysis" where the organization cannot arrive at a meaningful conclusion to the effort. For long-term, large-scale transformations, quick sprints are not necessary and can be counterproductive.	The waterfall approach is favorable in this situation. For strategies to be developed, a structured sequence of activities are required to arrive at a decision point: current state assessment --> future state goals / requirements definition -- > future state recommendations. Each of these steps has dependencies on the one preceding.
Analysis	In some cases, it makes sense to apply an agile approach to analysis. Insights can be shared with stakeholders as they are uncovered, rather than in a single final presentation. Based on ongoing feedback, the analysis can continuously be updated and tweaked to test new hypothesis.	In a waterfall model, findings would be summarized, written down, and later reviewed/validated with key stakeholders all at once along with critical takeaways. Both options can be viable depending on size and complexity of the analysis at hand.
Proof of Concept	POCs are unpredictable in nature, making an agile approach preferrable over waterfall. During POCs, teams often need to reassess and pivot their approach as both sides develop a deeper understanding of the problem statement and the solution options	A waterfall approach is not favorable for a PoC as it does not provide enough flexibility to adapt to new learnings. The crucial dependency on iterative feedback makes the waterfall approach a higher risk in a POC effort.
Implementation	Implementations are the lone example on this list that should be considered as either Agile, Waterfall, or potentially a hybrid of both approaches.	



Agile and Waterfall in Implementations

The Waterfall methodology carries significant risks in the early phases of a project, particularly around requirements gathering, application development, and integration development. Because each phase is completed in its entirety, inadequate or misunderstood requirements can lead to costly rework and delays later in the development process, especially if issues are only discovered during integration. In contrast, Agile methodology shifts the primary risks to later stages, where frequent changes and iterative development can result in insufficient time or attention for thorough testing, end-user training, and the development of robust standard operating procedures (SOPs), potentially impacting deployment readiness and long-term usability.

Supply chain projects of any category carry risk. They require time, effort, and money and have high expectations of return on such investment. Whether executing on a new organizational strategy following a market shift, committing to an analysis of transactional data, or standing up new software – all require collective effort among many individuals both inside and outside an organization to hit a moving target. It should be expected there will be challenges along the way. Without alignment on an approach, organizations are likely to fail to meet those challenges. Conventional, Agile, or some creative combination of methodologies can all be viable as long as there is a realistic understanding of where the risks lie with the approach.

Key Takeaways

- **Projects are about people.** Methods aren't formulas you plug numbers into to get a guaranteed answer. People are at the center of getting things done.
- **Find the right balance.** Good project teams know when to stick to structure and when to stay flexible. That balance often comes from working through different opinions to land on what fits best for the project.
- **Write it down, share it, get feedback.** Talking it out in a meeting isn't enough to align on an approach. Put your methodology in writing, share it with the team, and encourage (or require) review and sign-off so everyone's on the same page.
- **Be upfront about the risks.** No method will magically fix everything. Be clear about where the organization might struggle and what's realistic, then manage expectations and work to reduce risks where you can.

About Forerunners

Forerunners Consulting delivers specialized and focused consulting services, solving today's most challenging supply chain and transportation opportunities. Our differentiated and targeted approach is what sets us apart; we execute strategy, analytics, and implementation projects with an energy that inspires and builds momentum for our clients and trust in our ability to execute. We are made up of a team of strategic, technical, and experienced consultants that are skilled in delivering value and a quick return on investment. The way we approach problems is a mindset, and we challenge our clients to move them towards impactful change. To learn more about Forerunners, visit www.forerunnersconsulting.com.